

Inference algorithms for pattern-based CRFs on sequence data

Rustem Takhanov*

Institute of Science and Technology (IST), Austria

TAKHANOV@MAIL.RU

Vladimir Kolmogorov*

Institute of Science and Technology (IST), Austria

VNK@IST.AC.AT

Abstract

We consider *Conditional Random Fields (CRFs) with pattern-based potentials* defined on a chain. In this model the energy of a string (labeling) $x_1 \dots x_n$ is the sum of terms over intervals $[i, j]$ where each term is non-zero only if the substring $x_i \dots x_j$ equals a prespecified pattern α . Such CRFs can be naturally applied to many sequence tagging problems.

We present efficient algorithms for the three standard inference tasks in a CRF, namely computing (i) the partition function, (ii) marginals, and (iii) computing the MAP. Their complexities are respectively $O(nL)$, $O(nL\ell_{\max})$ and $O(nL \min\{|D|, \log(\ell_{\max}+1)\})$ where L is the combined length of input patterns, ℓ_{\max} is the maximum length of a pattern, and D is the input alphabet. This improves on the previous algorithms of (Ye et al., 2009) whose complexities are respectively $O(nL|D|)$, $O(n|\Gamma|L^2\ell_{\max}^2)$ and $O(nL|D|)$, where $|\Gamma|$ is the number of input patterns.

In addition, we give an efficient algorithm for sampling. Finally, we consider the case of non-positive weights. (Komodakis & Paragios, 2009) gave an $O(nL)$ algorithm for computing the MAP. We present a modification that has the same worst-case complexity but can beat it in the best case.

1. Introduction

This paper addresses the *sequence labeling* (or the *sequence tagging*) problem: given an observation z

(which is usually a sequence of n values), infer labeling $x = x_1 \dots x_n$ where each variable x_i takes values in some finite domain D . Such problem appears in many domains such as text and speech analysis, signal analysis, and bioinformatics.

One of the most successful approaches for tackling the problem is the Hidden Markov Model (HMM). The k th order HMM is given by the probability distribution $p(x|z) = \frac{1}{Z} \exp\{-E(x|z)\}$ with the energy function

$$E(x|z) = \sum_{i \in [1, n]} \psi_i(x_i, z_i) + \sum_{(i, j) \in \mathcal{E}_k} \psi_{ij}(x_{i:j}) \quad (1)$$

where $\mathcal{E}_k = \{(i, i+k) | i \in [1, n-k]\}$ and $x_{i:j} = x_i \dots x_j$ is the substring of x from i to j . A popular generalization is the Conditional Random Field model (Lafferty et al., 2001) that allows all terms to depend on the full observation z :

$$E(x|z) = \sum_{i \in [1, n]} \psi_i(x_i, z) + \sum_{(i, j) \in \mathcal{E}_k} \psi_{ij}(x_{i:j}, z) \quad (2)$$

We study a particular variant of this model called a *pattern-based CRF*. It is defined via

$$E(x|z) = \sum_{\alpha \in \Gamma} \sum_{\substack{[i, j] \subseteq [1, n] \\ j-i+1=|\alpha|}} \psi_{ij}^\alpha(z) \cdot [x_{i:j} = \alpha] \quad (3)$$

where Γ is a fixed set of non-empty words, $|\alpha|$ is the length of word α and $[\cdot]$ is the *Iverson bracket*. If we take $\Gamma = D \cup D^k$ then (3) becomes equivalent to (2); thus we do not lose generality (but gain more flexibility).

Intuitively, pattern-based CRFs allow to model long-range interactions for selected subsequences of labels. This could be useful for a variety of applications: in part-of-speech tagging patterns could correspond to certain syntactic constructions or stable idioms; in protein secondary structure prediction - to sequences of dihedral angles corresponding to stable configuration such as α -helices; in gene prediction - to sequences

* The authors contributed equally to the work, and assert joint first authorship.

of nucleatydies with supposed functional roles such as “exon” or “intron”, specific codons, etc.

Inference This paper focuses on inference algorithms for pattern-based CRFs. The three standard inference tasks are (i) computing the partition function $Z = \sum_x \exp\{-E(x|z)\}$; (ii) computing marginal probabilities $p(x_{i:j} = \alpha|z)$ for all triplets (i, j, α) present in (3); (iii) computing MAP, i.e. minimizing energy (3). The complexity of solving these tasks is discussed below. We denote $L = \sum_{\alpha \in \Gamma} |\alpha|$ to be total length of patterns and $\ell_{\max} = \max_{\alpha \in \Gamma} |\alpha|$ to be the maximum length of a pattern.

A naive approach is to use standard message passing techniques for an HMM of order $k = \ell_{\max} - 1$. However, they would take $O(n|D|^{k+1})$ time which would become impractical for large k . More efficient algorithms with complexities $O(nL|D|)$, $O(n|\Gamma|L^2\ell_{\max}^2)$ and $O(nL|D|)$ respectively were given by (Ye et al., 2009).¹ Our first contribution is to improve this to $O(nL)$, $O(nL\ell_{\max})$ and $O(nL \cdot \min\{|D|, \log(\ell_{\max} + 1)\})$ respectively (more accurate estimates are given in the next section).

We also give an algorithm for sampling from the distribution $p(x|z)$. Its complexity is either (i) $O(nL)$ per sample, or (ii) $O(n)$ per sample with an $O(nL|D|)$ preprocessing (assuming that we have an oracle that produces independent samples from the uniform distribution on $[0, 1]$ in $O(1)$ time).

Finally, consider the case when all costs $\psi_{ij}^\alpha(z)$ are non-positive. (Komodakis & Paragios, 2009) gave an $O(nL)$ technique for minimizing energy (3) in this case. We present a modification that has the same worst-case complexity but can beat the algorithm of (Komodakis & Paragios, 2009) in the best case.

Related work The works of (Ye et al., 2009) and (Komodakis & Paragios, 2009) mentioned above are probably the most related to our paper. The former applied pattern-based CRFs to the handwritten character recognition problem and to the problem of identification of named entities from texts. The latter considered a pattern-based CRF on a grid for a computer vision application; the MAP inference problem in (Komodakis & Paragios, 2009) was converted to sequence labeling problems by decomposing the grid into thin “stripes”.

(Qian et al., 2009) considered a more general formulation in which a single pattern is characterized by a set of strings rather than a single string α . They pro-

posed an exact inference algorithm and applied it to the OCR task and to the Chinese Organization Name Recognition task. However, their algorithm could take time exponential in the total lengths of input patterns; no subclasses of inputs were identified which could be solved in polynomial time.

A different generalization (for non-sequence data) was proposed in (Rother et al., 2009). Their inference procedure reduces the problem to the MAP estimation in a pairwise CRF with cycles, which is then solved with approximate techniques such as BP, TRW or QPBO. This model was applied to the texture restoration problem.

(Nguyen et al., 2011) extended algorithms in (Ye et al., 2009) to the *Semi-Markov model* (Sarawagi & Cohen, 2004). We conjecture that our algorithms can be extended to this case as well, and can yield a better complexity compared to (Nguyen et al., 2011).

2. Notation and preliminaries

First, we introduce a few definitions.

- A *pattern* is a pair $\alpha = ([i, j], x)$ where $[i, j]$ is an interval in $[1, n]$ and $x = x_i \dots x_j$ is a sequence over alphabet D indexed by integers in $[i, j]$ ($j \geq i - 1$). The *length* of α is denoted as $|\alpha| = |x| = j - i + 1$.
- Symbols “*” denotes an arbitrary word or pattern (possibly the empty word ε or the empty pattern $\varepsilon_s \triangleq ([s + 1, s], \varepsilon)$ at position s). The exact meaning will always be clear from the context. Similary, “+” denotes an arbitrary non-empty word or pattern.
- The concatenation of patterns $\alpha = ([i, j], x)$ and $\beta = ([j + 1, k], y)$ is the pattern $\alpha\beta \triangleq ([i, k], xy)$. Whenever we write $\alpha\beta$ we assume that it is defined, i.e. $\alpha = ([\cdot, j], \cdot)$ and $\beta = ([j + 1, \cdot], \cdot)$ for some j .
- For a pattern $\alpha = ([i, j], x)$ and interval $[k, \ell] \subseteq [i, j]$, the *subpattern* of α at position $[k, \ell]$ is the pattern $\alpha_{k:\ell} \triangleq ([k, \ell], x_{k:\ell})$ where $x_{k:\ell} = x_k \dots x_\ell$. If $k = i$ then $\alpha_{k:\ell}$ is called a *prefix* of α . If $\ell = j$ then $\alpha_{k:\ell}$ is a *suffix* of α .
- If β is a subpattern of α , i.e. $\beta = \alpha_{k:\ell}$ for some $[k, \ell]$, then we say that β is *contained* in α . This is equivalent to the condition $\alpha = *\beta*$.
- $D^{i:j} = \{([i, j], x) \mid x \in D^{[i, j]}\}$ is the set of patterns with interval $[i, j]$. We typically use letter x for patterns in $D^{1:s}$ and letters α, β, \dots for other patterns. Patterns $x \in D^{1:s}$ will be called *partial labelings*.
- For a set of patterns Π and index $s \in [0, n]$ we denote Π_s to be the set of patterns in Π that end at position s : $\Pi_s = \{([i, s], \alpha) \in \Pi\}$.
- For a pattern α let α^- be the prefix of α of length $|\alpha| - 1$; if α is empty then α^- is undefined.

¹Some of the bounds stated in (Ye et al., 2009) are actually weaker. However, it is not difficult to show that their algorithms can be implemented in times stated above, using our Lemma 1.

efficiently.

Lemma 1. *Let Π be a set of patterns with $\varepsilon_s \in \Pi$ for all $s \in [0, n]$. Values $\phi(\alpha)$ for all $\alpha \in \Pi$ can be computed using $O(|\Pi|)$ multiplications (“ \otimes ”). The same holds for values $f(\alpha)$ assuming that Π is prefix-closed, i.e. $\alpha^- \in \Pi$ for all non-empty patterns $\alpha \in \Pi$.*

Proof. To compute $\phi(\cdot)$ for patterns $\alpha \in \Pi_s$, we use the following procedure: (i) set $\phi(\varepsilon_s) := \mathbb{1}$; (ii) traverse edges $(\alpha, \beta) \in E[\Pi_s]$ of tree $G[\Pi_s]$ (from the root to the leaves) and set

$$\phi(\beta) := \begin{cases} \phi(\alpha) \otimes c_\beta & \text{if } \beta \in \Pi^\circ \\ \phi(\alpha) & \text{otherwise} \end{cases}$$

Now suppose that Π is prefix-closed. After computing $\phi(\cdot)$, we go through indexes $s \in [0, n]$ and set

$$f(\varepsilon_s) := \mathbb{1}, \quad f(\alpha) := f(\alpha^-) \otimes \phi(\alpha) \quad \forall \alpha \in \Pi_s - \{\varepsilon_s\}$$

□

Sets of partial labelings Let Π_s be a set of patterns that end at position s . Assume that $\varepsilon_s \in \Pi_s$. For a pattern $\alpha \in \Pi_s$ we define

$$\mathcal{X}_s(\alpha) = \{x \in D^{1:s} \mid x = *\alpha\} \quad (7)$$

$$\mathcal{X}_s(\alpha; \Pi_s) = \mathcal{X}_s(\alpha) - \bigcup_{(\alpha, \beta) \in E[\Pi_s]} \mathcal{X}_s(\beta) \quad (8)$$

It can be seen that sets $\mathcal{X}_s(\alpha; \Pi_s)$ are disjoint, and their union over $\alpha \in \Pi_s$ is $D^{1:s}$. Furthermore, there holds

$$\mathcal{X}_s(\alpha; \Pi_s) = \{x \in \mathcal{X}_s(\alpha) \mid x \neq *\beta \quad \forall \beta = +\alpha \in \Pi_s\} \quad (9)$$

We will use eq. (9) as the definition of $\mathcal{X}_s(\alpha; \Pi_s)$ in the case when $\alpha \notin \Pi_s$.

3. Computing partition function

In this section we give an algorithm for computing quantity (5) assuming that (R, \oplus, \otimes) is a ring. This can be used, in particular, for computing the partition function. We will assume that $D \subseteq \Gamma$; we can always add D to Γ if needed².

²Note that we still claim complexity $O(nP)$ where P is the number of distinct non-empty prefixes of words in the original set Γ . Indeed, we can assume w.l.o.g. that each letter in D occurs in at least one word $\alpha \in \Gamma$. (If not, then we can “merge” non-occurring letters to a single letter and add this letter to Γ ; clearly, any instance over the original pair (D, Γ) can be equivalently formulated as an instance over the new pair. The transformation increases P only by 1). The assumption implies that $|D| \leq P$. Adding D to Γ increases P by at most P , and thus does not affect bound $O(nP)$.

First, we select set Π as the set of prefixes of patterns in Π° :

$$\Pi = \{\alpha \mid \exists \alpha^* \in \Pi^\circ\} \quad (10)$$

We will compute the following quantities for each $s \in [0, n]$, $\alpha \in \Pi_s$:

$$M_s(\alpha) = \bigoplus_{x \in \mathcal{X}_s(\alpha; \Pi_s)} f(x), \quad W_s(\alpha) = \bigoplus_{x \in \mathcal{X}_s(\alpha)} f(x) \quad (11)$$

It is easy to see that for $\alpha \in \Pi_s$ the following equalities relate M_s and W_s :

$$M_s(\alpha) = W_s(\alpha) \ominus \bigoplus_{(\alpha, \beta) \in E[\Pi_s]} W_s(\beta) \quad (12a)$$

$$W_s(\alpha) = M_s(\alpha) \oplus \bigoplus_{(\alpha, \beta) \in E[\Pi_s]} W_s(\beta) \quad (12b)$$

These relations motivate the following algorithm. Since $|\Pi_s| = P + 1$ for indexes s that are sufficiently far from the boundary, its complexity is $\Theta(nP)$ assuming that values $\phi(\alpha)$ in eq. (13a) are computed using Lemma 1.

Algorithm 1 Computing $Z = \bigoplus_{x \in D^{1:n}} f(x)$ for a ring

- 1: initialize messages: set $W_0(\varepsilon_0) := \mathbb{O}$
- 2: for each $s = 1, \dots, n$ traverse nodes $\alpha \in \Pi_s$ of tree $G[\Pi_s]$ starting from the leaves and set

$$M_s(\alpha) := \phi(\alpha) \otimes \left[W_{s-1}(\alpha^-) \ominus \bigoplus_{(\alpha, \beta) \in E[\Pi_s]} W_{s-1}(\beta^-) \right] \quad (13a)$$

$$W_s(\alpha) := M_s(\alpha) \oplus \bigoplus_{(\alpha, \beta) \in E[\Pi_s]} W_s(\beta) \quad (13b)$$

Exception: if $\alpha = \varepsilon_s$ then set $M_s(\alpha) := \mathbb{O}$ instead of (13a)

- 3: return $Z := W_n(\varepsilon_n)$
-

Theorem 2. *Algorithm 1 is correct, i.e. it returns the correct value of $Z = \bigoplus_x f(x)$.*

3.1. Proof of Theorem 2

Eq. (13b) coincides with (12b); let us show that eq. (13a) holds for any $\alpha \in \Pi_s - \{\varepsilon_s\}$. (Note, for $\alpha = \varepsilon_s$ step 2 is correct: assumption $D \subseteq \Gamma$ implies that $D^{s:s} \subseteq \Pi_s$, and therefore $\mathcal{X}_s(\varepsilon_s; \Pi_s) = \emptyset$, $M_s(\varepsilon_s) = \mathbb{O}$).

For a partial labeling $x \in D^{1:s}$ define the “reduced partial cost” as

$$f^-(x) = \bigotimes_{\alpha \in \Pi^\circ, x = *\alpha} c_\alpha \quad (14)$$

It is easy to see from (11) that for any $\alpha \in \Pi_s - \{\varepsilon_s\}$

$$W_{s-1}(\alpha^-) = \sum_{x \in \mathcal{X}_s(\alpha)} f^-(x) \quad (15)$$

Consider $\alpha \in \Pi_s - \{\varepsilon_s\}$. We will show that for any $x \in \mathcal{X}_s(\alpha)$ there holds

$$\llbracket x \in \mathcal{X}_s(\alpha; \Pi_s) \rrbracket \otimes f(x) = \phi(\alpha) \otimes \left[f^-(x) \ominus \bigoplus_{\substack{(\alpha, \beta) \in E[\Pi_s]: \\ x \in \mathcal{X}_s(\beta)}} f^-(x) \right] \quad (16)$$

where $\llbracket \cdot \rrbracket = \mathbb{1}$ if the argument is true, and \ominus otherwise. This will be sufficient for establishing the theorem: summing these equations over $x \in \mathcal{X}_s(\alpha)$ and using (11), (15) yields eq. (13a).

Two cases are possible:

Case 1: $x \in \mathcal{X}_s(\beta)$ for some $(\alpha, \beta) \in E[\Pi_s]$. (Such β is unique since sets $\mathcal{X}_s(\beta)$ are disjoint.) Then both sides of (16) are \ominus .

Case 2: $x \in \mathcal{X}_s(\alpha; \Pi_s)$. Then eq. (16) is equivalent to $f(x) = \phi(\alpha) \otimes f^-(x)$. This holds since there is no pattern $\gamma \in \Pi_s^\circ(x)$ with $|\gamma| > |\alpha|$ (otherwise we would have $\gamma \in \Pi_s$ and thus $x \notin \mathcal{X}_s(\alpha; \Pi_s)$ by definition (9) - a contradiction).

4. Sampling

In this section consider the semiring $(R, \oplus, \otimes) = (\mathbb{R}, +, \times)$ from Example 1. We assume that all costs c_α are strictly positive. We present an algorithm for sampling labelings $x \in D^{1:n}$ according to the probability distribution $p(x) = f(x)/Z$.

As in the previous section, we assume that $D \subseteq \Gamma$, and define Π to be the set of prefixes of patterns in Π° (eq. (10)). For a node $\alpha \in \Pi_s$ let $T_s(\alpha)$ be the set of nodes in the subtree of $G[\Pi_s]$ rooted at α , with $\alpha \in T_s(\alpha) \subseteq \Pi_s$. For a pattern $\alpha \in \Pi_{s+1} - \{\varepsilon_{s+1}\}$ we define set

$$\Delta_s(\alpha) = T_s(\alpha^-) - \bigcup_{(\alpha, \beta) \in G[\Pi_{s+1}]} T_s(\beta^-) \quad (17)$$

We can now present the algorithm.

Algorithm 2 Sample $x \sim p(x) = f(x)/Z$

- 1: run Algorithm 1 to compute messages $M_s(\alpha)$ for all patterns $\alpha = ([\cdot, s], \cdot) \in \Pi$
 - 2: sample $\alpha_n \in \Pi_n$ with probability $p(\alpha_n) \propto M_n(\alpha_n)$
 - 3: for $s = n-1, \dots, 1$ sample $\alpha_s \in \Delta_s(\alpha_{s+1})$ with probability $p(\alpha_s) \propto M_s(\alpha_s)$
 - 4: return labeling x with $x_{s:s} = (\alpha_s)_{s:s}$ for $s \in [1, n]$
-

We say that step s of the algorithm is *valid* if either (i) $s = n$, or (ii) $s \in [1, n-1]$, step $s+1$ is valid, $\alpha_{s+1} \neq \varepsilon_{s+1}$ and $M_s(\alpha) > 0$ for some $\alpha \in \Delta_s(\alpha_{s+1})$. (This is a recursive definition.) Clearly, if step s is valid then line 3 of the algorithm is well-defined.

Theorem 3. (a) With probability 1 all steps of the algorithm are valid. (b) The returned labeling $x \in D^{1:n}$ is distributed according to $p(x) = f(x)/Z$.

Complexity Assume that we have an oracle that produces independent samples from the uniform distribution on $[0, 1]$ in $O(1)$ time.

The main subroutine performed by the algorithm is sampling from a given discrete distribution. Clearly, this can be done in $\Theta(N)$ time where N is the number of allowed values of the random variable. With a $\Theta(N)$ preprocessing, a sample can also be produced in $O(1)$ time by the so-called “alias method” (Vose, 1991).

This leads to two possible complexities: (i) $\Theta(nP)$ (without preprocessing); (ii) $\Theta(n)$ per sample (with preprocessing). Let us discuss the complexity of this preprocessing. Running Algorithm 1 takes $\Theta(nP)$ time. After that, for each $\alpha \in \Pi_{s+1}$ we need to run the linear time procedure of (Vose, 1991) for distributions $p(\beta) \propto M_s(\beta), \beta \in \Delta_s(\alpha_{s+1})$. The following theorem implies that this takes $\Theta(nP|D|)$ time.

Theorem 4. There holds $\sum_{\alpha \in \Pi_s - \{\varepsilon_s\}} |\Delta_{s-1}(\alpha)| = |\Pi_{s-1}| \cdot |D|$.

Proof. Consider pattern $\beta \in \Pi_{s-1}$. For a letter $a \in D^{s:s}$ let $\beta^a \in \Pi_s$ be longest suffix α of βa with $\alpha \in \Pi_s$ (at least one such suffix exists, namely a). It can be seen that the set $\{\beta^a \mid a \in D^{s:s}\}$ is exactly the set of patterns $\alpha \in \Pi_s - \{\varepsilon_s\}$ for which $\Delta_{s-1}(\alpha)$ contains β (checking this fact is just definition chasing). Therefore, the sum in the theorem equals $\sum_{\beta \in \Pi_{s-1}} |\{\beta^a \mid a \in D^{s:s}\}| = |\Pi_{s-1}| \cdot |D|$. \square

To summarize, we showed that with a $\Theta(nP|D|)$ preprocessing we can compute independent samples from $p(x)$ in $\Theta(n)$ time per sample.

4.1. Proof of Theorem 3

Suppose that step $s \in [1, n]$ of the algorithm is valid; this means that patterns α_t for $t \in [s, n]$ are well-defined. For $t \in [s, n]$ we then define the set of patterns $\mathcal{A}_t = \Delta_t(\alpha_{t+1}) \subseteq \Pi_t$ (if $t = n$ then we define $\mathcal{A}_t = \Pi_t$ instead). We also define sets of labelings

$$\mathcal{Y}_t(\alpha) = \{yx_{t+1:n} \mid y \in \mathcal{X}_t(\alpha; \Pi_t)\} \quad \forall \alpha \in \mathcal{A}_t \quad (18)$$

$$\mathcal{Y}_t = \mathcal{Y}_t(\alpha_t) \quad (19)$$

where x is a labeling with $x_{t:t} = (\alpha_t)_{t:t}$ for $t \in [s, n]$. Let $\mathcal{Y}_{n+1} = D^{1:n}$.

Lemma 5. *Suppose that step $s \in [1, n]$ is valid.*

(a) \mathcal{Y}_{s+1} is a disjoint union of sets $\mathcal{Y}_s(\alpha)$ over $\alpha \in \mathcal{A}_s$.
 (b) For each $y \in \mathcal{Y}_{s+1} = \bigcup_{\alpha \in \mathcal{A}_s} \mathcal{Y}_s(\alpha)$ there holds $f(y) = \text{const}_s \cdot f(y_{1:s})$, and consequently for any $\alpha \in \mathcal{A}_s$ there holds

$$\sum_{y \in \mathcal{Y}_s(\alpha)} f(y) = \text{const}_s \cdot \sum_{y \in \mathcal{X}_s(\alpha; \Pi_s)} f(y) = \text{const}_s \cdot M_s(\alpha)$$

Theorem 3 will follow from this lemma. Indeed, the lemma shows that the algorithm implicitly computes a sequence of nested sets $D^{1:n} = \mathcal{Y}_{n+1} \supseteq \mathcal{Y}_n \supseteq \dots \supseteq \mathcal{Y}_1 = \{x\}$. At step s we divide set \mathcal{Y}_{s+1} into disjoint subsets $\mathcal{Y}_s(\alpha)$, $\alpha \in \mathcal{A}_s$ and select one of them, $\mathcal{Y}_s = \mathcal{Y}_s(\alpha_s)$, with the probability proportional to $M_s(\alpha_s) \propto \sum_{y \in \mathcal{Y}_s(\alpha_s)} f(y)$.

We still need to show that if step $s \in [2, n]$ is valid then step $s - 1$ is valid as well with probability 1. It follows from the precondition that α_s sampled in line 3 satisfies $M_s(\alpha_s) > 0$ with probability 1; this implies that $\alpha_s \neq \varepsilon_s$. From the paragraph above we get that $\sum_{y \in \mathcal{Y}_s} f(y) > 0$ with probability 1. We also have $\sum_{\alpha \in \mathcal{A}_{s-1}} M_{s-1}(\alpha) \propto \sum_{\alpha \in \mathcal{A}_{s-1}} \sum_{y \in \mathcal{Y}_{s-1}(\alpha)} f(y) = \sum_{y \in \mathcal{Y}_s} f(y) > 0$ implying that $M_{s-1}(\alpha) > 0$ for some $\alpha \in \mathcal{A}_{s-1}$. This concludes the proof that step $s - 1$ is valid with probability 1.

It remains to prove Lemma 5.

Part (a) First, we need to check that $\mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$ is equal to the disjoint union of $\mathcal{X}_s(\alpha; \Pi_s)$ over $\alpha \in \Delta_s(\alpha_{s+1})$ where $\Pi_{s+1}^- = \{\alpha^- \mid \alpha \in \Pi_{s+1}\}$. Disjointness of $\mathcal{X}_s(\alpha; \Pi_s)$ for different $\alpha \in \Pi_s$ is obvious. Since $\Pi_{s+1}^- \subseteq \Pi_s$, then for any $\alpha \in \Delta_s(\alpha_{s+1})$, $\mathcal{X}_s(\alpha; \Pi_s) \subseteq \mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$ is straightforward from the definition of $\Delta_s(\alpha_{s+1})$. Thus, we only need to check the inclusion of $\mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$ in the union.

Elements of $\Pi_{s+1}^- \cup \{\alpha_{s+1}^-\}$ can be seen as nodes in tree $G[\Pi_s]$. Then any pattern x from $\mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$ defines the longest suffix $s(x)$ such that $s(x) \in \Pi_s$. It is easy to see that $s(x) \in T_s(\alpha_{s+1}^-)$, and moreover, the descending path in $G[\Pi_s]$ from α_{s+1}^- to $s(x)$ does not contain elements from $\Pi_{s+1}^- - \{\alpha_{s+1}^-\}$, otherwise $x, s(x) \notin \mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$. It is easy to see that this is equivalent to $s(x) \in \Delta_s(\alpha_{s+1})$. Since $x \in \mathcal{X}_s(s(x); \Pi_s)$, $\mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)$ is a subset of the union of $\mathcal{X}_s(\alpha; \Pi_s)$ over $\alpha \in \Delta_s(\alpha_{s+1})$.

Now according to definition of \mathcal{Y}_{s+1} we can write:

$$\begin{aligned} \mathcal{Y}_{s+1} &= \{yx_{s+2:n} \mid y \in \mathcal{X}_{s+1}(\alpha_{s+1}; \Pi_{s+1})\} = \\ &= \{y(\alpha_{s+1})_{s+1:s+1}x_{s+2:n} \mid y \in \mathcal{X}_s(\alpha_{s+1}^-; \Pi_{s+1}^-)\} = \\ &= \bigcup_{\alpha \in \Delta_s(\alpha_{s+1})} \{y(\alpha_{s+1})_{s+1:s+1}x_{s+2:n} \mid y \in \mathcal{X}_s(\alpha; \Pi_s)\} \end{aligned}$$

It only remains to check that in the last union the set corresponding to $\alpha \in \Delta_s(\alpha_{s+1})$ is exactly equal to $\mathcal{Y}_s(\alpha)$.

Part (b) Let p be the start position of α_{s+1} , i.e. $\alpha_{s+1} = ([p, s+1], \cdot)$. Consider labeling $y \in \mathcal{Y}_{s+1}$, we then must have $y = *\alpha_{s+1}*$. Let $\beta = ([i, j], \cdot)$ be a pattern with $y = *\beta*$, $j > s$. We will prove that $i \geq p$; this will imply the claim.

Suppose on the contrary that $i < p$. Denote $\gamma = \beta_{i:s+1}$, then $\gamma \in \Pi_{s+1}$ and $\gamma = +\alpha_{s+1}$. Therefore, $y_{1:s+1} \notin \mathcal{X}_{s+1}(\alpha_{s+1}; \Pi_{s+1})$ (since $y_{1:s+1} = *\gamma$). However, this contradicts the assumption that $y \in \mathcal{Y}_{s+1} = \mathcal{X}_{s+1}(\alpha_{s+1})$.

5. Computing marginals

In this section we again consider the semiring $(R, \oplus, \otimes) = (\mathbb{R}, +, \times)$ from Example 1 where all costs c_α are strictly positive, and consider a probability distribution $p(x) = f(x)/Z$ over labelings $x \in D^{1:n}$.

For a pattern α we define

$$\Omega(\alpha) = \{x \in D^{1:n} \mid x = *\alpha*\} \quad (20)$$

$$Z(\alpha) = \sum_{x \in \Omega(\alpha)} f(x) \quad (21)$$

We also define the set of patterns

$$\Pi = \{\alpha \mid \exists \alpha^*, *\alpha \in \Pi^\circ, \alpha \text{ is non-empty}\} \quad (22)$$

Note that $\Pi^\circ \subseteq \Pi$ and $|\Pi_s| = |I(\Gamma)|$ for indexes s that are sufficiently far from the boundary. We will present an algorithm for computing $Z(\alpha)$ for all patterns $\alpha \in \Pi$ in time $O(n \sum_{\alpha \in I(\Gamma)} |\alpha|)$. Marginal probabilities of a pattern-based CRF can then be computed as $p(x_{i:j} = \alpha) = Z(\alpha)/Z$ for a pattern $\alpha = ([i, j], \cdot)$.

In the previous section we used graph $G[\Pi_s]$ for a set of patterns Π_s ; here we will need an analogous but a slightly different construction for patterns in Π . For patterns α, β we write $\alpha \sqsubseteq \beta$ if $\beta = *\alpha*$. If we have $\beta = +\alpha+$ then we write $\alpha \sqsubset \beta$.

Now consider $\alpha \in \Pi$. We define $\Phi(\alpha)$ to be the set of patterns $\beta \in \Pi$ such that $\alpha \sqsubset \beta$ and there is no other pattern $\gamma \in \Pi$ with $\alpha \sqsubset \gamma \sqsubseteq \beta$.

Our algorithm is given below. In the first step it runs Algorithm 1 from left to right and from right to left; as

a result, we get forward messages $\vec{W}_j(\alpha)$ and backward messages $\overleftarrow{W}_i(\alpha)$ for patterns $\alpha = ([i, j], \cdot)$ such that

$$\vec{W}_j(\alpha) = \sum_{\substack{x=\alpha \\ x \in D^{1:j}}} f(x) \quad \overleftarrow{W}_i(\alpha) = \sum_{\substack{y=\alpha^* \\ y \in D^{i:n}}} f(y) \quad (23)$$

Algorithm 3 Computing values $Z(\alpha)$

1: run Algorithm 1 in both directions to get messages $\vec{W}_j(\alpha), \overleftarrow{W}_i(\alpha)$. For each pattern $\alpha = ([i, j], \cdot) \in \Pi$ set

$$W(\alpha) := \frac{\vec{W}_j(\alpha) \overleftarrow{W}_i(\alpha)}{f(\alpha)} \quad (24a)$$

$$W^-(\alpha) := \frac{\vec{W}_{j-1}(\alpha_{i:j-1}) \overleftarrow{W}_{i+1}(\alpha_{i+1:j})}{f(\alpha_{i+1:j-1})} \quad (24b)$$

2: for $\alpha \in \Pi$ (in the order of decreasing $|\alpha|$) set

$$Z(\alpha) := W(\alpha) + \sum_{\beta \in \Phi(\alpha)} [Z(\beta) - W^-(\beta)] \quad (25)$$

Theorem 6. *Algorithm 3 is correct.*

We prove the theorem in section 5.1, but first let us discuss algorithm's complexity. We claim that all values $f(\alpha)$ used by the algorithm can be computed in $O(n(P+S))$ time where P and S are respectively the number of distinct non-empty prefixes and suffixes of words in Γ . Indeed, we first compute these values for patterns in the set $\vec{\Pi} \triangleq \{\alpha \mid \exists \alpha^* \in \Pi^\circ\}$; by Lemma 1, this takes $O(nP)$ time. This covers values $f(\alpha)$ used in eq. (24a). As for the value in eq. (24b) for pattern $\alpha = ([i, j], \cdot) \in \Pi$, we can use the formula

$$f(\alpha_{i+1:j-1}) = \frac{f(\alpha) \tilde{c}_\alpha}{\vec{\phi}(\alpha) \overleftarrow{\phi}(\alpha)}$$

where $\tilde{c}_\alpha = c_\alpha$ if $\alpha \in \Pi^\circ$ and $\tilde{c}_\alpha = 1$ otherwise, and

$$\vec{\phi}(\alpha) = \prod_{\beta \in \Pi^\circ, \alpha = *\beta} c_\beta, \quad \overleftarrow{\phi}(\alpha) = \prod_{\beta \in \Pi^\circ, \alpha = \beta^*} c_\beta$$

The latter values can be computed in $O(n(P+S))$ time by applying Lemma 1 in the forward and backward directions. (In fact, there were already computed when running Algorithm 1.)

We showed that step 1 can be implemented in $O(n(P+S))$ time; let us analyze step 2. The following lemma implies that it performs $O(n \sum_{\alpha \in I(\Gamma)} |\alpha|)$ arithmetic operations; since $\sum_{\alpha \in I(\Gamma)} |\alpha| \geq \sum_{\alpha \in \Gamma} |\alpha| \geq \max\{P, S\}$, we then get that the overall complexity is $O(n \sum_{\alpha \in I(\Gamma)} |\alpha|)$.

Lemma 7. *For each $\beta \in \Pi$ there exist at most $2|\beta|$ patterns $\alpha \in \Pi$ such that $\beta \in \Phi(\alpha)$.*

Proof. Let Ψ be the set of such patterns α . Note, there holds $\beta = +\alpha+$. We need to show that $m \triangleq |\Psi| \leq 2|\beta|$. Let us order patterns $\alpha = ([i, j], \cdot) \in \Psi$ lexicographically (first by i , then by j): $\Psi = \{\alpha_1, \dots, \alpha_m\}$ with $\alpha_t = ([i_t, j_t], \cdot)$, and denote $\sigma_t = (i_t - k) + (j_t - k) \in [2, 2(\ell - k - 1)]$ where $[k, \ell]$ is the interval for β . We will prove by induction that $\sigma_t \geq t + 1$ for $t \in [1, m]$; this will imply that $m + 1 \leq \sigma_m \leq 2(\ell - k - 1) = 2(|\beta| - 2)$, as desired.

The base case is trivial. Suppose that it holds for $t - 1$; let us prove it for t . If $i_t = i_{t-1}$ then $j_t > j_{t-1}$ by the definition of the order on Ψ , so the claim holds. Suppose that $i_t > i_{t-1}$. If $j_t < j_{t-1}$ then $\alpha_t \sqsubset \alpha_{t-1} \sqsubset \beta$ contradicting the condition $\beta \in \Phi(\alpha_t)$. Thus, $j_t \geq j_{t-1}$, and so the claim of the induction step holds. \square

Remark 1. An alternative method for computing marginals with complexity $O(n|\Gamma|L^2\ell_{\max}^2)$ was given in (Ye et al., 2009). They compute value $Z(\alpha)$ directly from messages $\vec{M}_{j'}(\cdot)$ and $\overleftarrow{M}_{i'}(\cdot)$ by summing over pairs of patterns (thus the square factor in the complexity). In contrast, we use a recursive rule that uses previously computed values of $Z(\cdot)$. We also use the existence of the “ \ominus ” operation. This allows us to achieve better complexity.

5.1. Proof of theorem 6

Consider labeling $x \in D^{1:n}$. We define $\Lambda(x) = \{\alpha \in \Pi^\circ \mid x = *\alpha*\}$ to be the set of patterns contained in x . For an interval $[i, j] \subseteq [1, n]$ we also define sets

$$\Lambda_{ij}(x) = \{\beta \in \Lambda(x) \mid \beta = +x_{i:j}+\} \quad (26a)$$

$$\Lambda_{ij}^-(x) = \{\beta \in \Lambda(x) \mid \beta = *x_{i:j}*\} \quad (26b)$$

and corresponding costs

$$f_{ij}(x) = \prod_{\beta \in \Lambda(x) - \Lambda_{ij}(x)} c_\beta \quad (27a)$$

$$f_{ij}^-(x) = \prod_{\beta \in \Lambda(x) - \Lambda_{ij}^-(x)} c_\beta \quad (27b)$$

It can be checked that quantities $W(\alpha)$, $W^-(\alpha)$ defined via (23) and (24) satisfy

$$W(\alpha) = \sum_{x \in \Omega(\alpha)} f_{ij}(x) \quad W^-(\alpha) = \sum_{x \in \Omega(\alpha)} f_{ij}^-(x) \quad (28)$$

where $[i, j]$ is the interval for α .

Consider pattern $\alpha = ([i, j], \cdot) \in \Pi$. We will show that for any $x \in \Omega(\alpha)$ there holds

$$f(x) = f_{ij}(x) + \sum_{\substack{\beta = ([k, \ell], \cdot) \in \Phi(\alpha): \\ x = *\beta*}} [f(x) - f_{k\ell}^-(x)] \quad (29)$$

This will be sufficient for establishing algorithm's correctness: summing these equations over $x \in \Omega(\alpha)$ and using (21),(28) yields eq. (25).

Lemma 8. *The sum in (29) contains at most one pattern $\beta = ([k, \ell], \cdot) \in \Phi(\alpha)$ with $x = *\beta*$.*

Proof. Consider two such patterns $\beta^1 = ([k^1, \ell^1], \cdot)$ and $\beta^2 = ([k^2, \ell^2], \cdot)$. Define $k = \max\{k^1, k^2\}$, $\ell = \min\{\ell^1, \ell^2\}$, $\beta = x_{k:\ell}$, then $\alpha \sqsubset \beta \sqsubseteq \beta^t$ for $t \in \{1, 2\}$. Using the definition of set Π , it can be checked that $\beta \in \Pi$. The fact that $\beta^t \in \Phi(\alpha)$ then implies that $\beta^t = \beta$ for $t \in \{1, 2\}$, and so $\beta^1 = \beta^2$. \square

We now consider two possible cases.

Case 1: There are no patterns $\beta = ([k, \ell], \cdot) \in \Phi(\alpha)$ with $x = *\beta*$. This implies that $\Lambda_{ij}(x)$ is empty, and therefore $f(x) = f_{ij}(x)$. Eq. (29) thus holds.

Case 2: There exists a (unique) pattern $\beta = ([k, \ell], \cdot) \in \Phi(\alpha)$ with $x = *\beta*$. Eq. (29) then becomes equivalent to the condition $f_{ij}(x) = f_{k\ell}^-(x)$. We will prove this by showing that $\Lambda_{ij}(x) = \Lambda_{k\ell}^-(x)$.

The inclusion $\Lambda_{k\ell}^-(x) \subseteq \Lambda_{ij}(x)$ is obvious; let us show the other direction. Suppose that $\gamma = ([p, q], \cdot) \in \Lambda_{ij}(x)$. Define $\hat{p} = \max\{k, p\}$, $\hat{q} = \min\{q, \ell\}$, $\hat{\gamma} = x_{\hat{p}:\hat{q}}$. It can be checked that $\hat{\gamma} \in \Pi$. We also have $\alpha \sqsubset \hat{\gamma} \sqsubseteq \beta$. Therefore, condition $\beta \in \Phi(\alpha)$ implies that $\hat{\gamma} = \beta$, and so $p \leq k$, $q \geq \ell$, and $\gamma \in \Lambda_{k\ell}^-(x)$.

6. General case: $O(nP'|D|)$ algorithm

In this section and in the next one we consider the case of a general commutative semiring (R, \oplus, \otimes) (without assuming the existence of an inverse operation for \oplus). This can be used for computing MAP in CRFs containing positive costs c_α . The algorithm closely resembles the method in (Ye et al., 2009); it is based on the same idea and has the same complexity. Our primary goal of presenting this algorithm is to motivate the $O(nP \log(\ell_{\max} + 1))$ algorithm for the MAP problem given in the next section.

First, we select Π as the set of proper prefixes of patterns in Π° :

$$\Pi = \{\alpha \mid \exists \alpha+ \in \Pi^\circ\} \quad (30)$$

For each $\alpha \in \Pi_s$ we will compute message

$$M_s(\alpha) = \bigoplus_{x \in \mathcal{X}_s(\alpha; \Pi_s)} f(x) \quad (31)$$

In order to go from step $s-1$ to s , we will use an extended set of patterns $\hat{\Pi}_s$:

$$\begin{aligned} \hat{\Pi}_s &= \{\alpha \mid \alpha^- \in \Pi_{s-1}\} \cup \{\varepsilon_s\} \\ &= \{\alpha c \mid \alpha \in \Pi_{s-1}, c \in D^{s:s}\} \cup \{\varepsilon_s\} \end{aligned} \quad (32)$$

It can be checked that

$$\Pi_s \subseteq \hat{\Pi}_s \quad \text{and} \quad \Pi_s^\circ \subseteq \hat{\Pi}_s \quad (33)$$

In step s we compute values $M_s(\alpha)$ in eq. (31) for all $\alpha \in \hat{\Pi}_s$. Note, we now use the generalized definition of $\mathcal{X}_s(\alpha; \Pi_s)$ (eq. (9)) since we may have $\alpha \notin \Pi_s$. After completing step s , messages $M_s(\alpha)$ for $\alpha \in \hat{\Pi}_s - \Pi_s$ can be discarded.

Our algorithm is given below. We have $|\hat{\Pi}_s| = P'|D| + 1$ for indexes s that are sufficiently far from the boundary, and thus the algorithm's complexity is $\Theta(nP'|D|)$ (if Lemma 1 is used for computing values $\phi(\alpha)$).

Algorithm 4 Computing $Z = \bigoplus_{x \in D^{1:n}} f(x)$

- 1: initialize messages: set $M_0(\varepsilon_0) := \mathbb{O}$
- 2: for each $s = 1, \dots, n$ traverse nodes $\alpha \in \hat{\Pi}_s$ of tree $G[\hat{\Pi}_s]$ starting from the leaves and set

$$M_s(\alpha) := [\phi(\alpha) \otimes M_{s-1}(\alpha^-)] \oplus \bigoplus_{(\alpha, \beta) \in E[\hat{\Pi}_s], \beta \notin \Pi_s} M_s(\beta) \quad (34)$$

If $\alpha = \varepsilon_s$ then use $M_{s-1}(\alpha^-) = \mathbb{O}$

- 3: return $Z := \bigoplus_{\alpha \in \Pi_n} M_n(\alpha)$
-

Theorem 9. *Algorithm 4 is correct.*

Remark 2 As we already mentioned, Algorithm 4 resembles the algorithm in (Ye et al., 2009). The latter computes the same set of messages as we do but using the following recursion: for a pattern $\alpha \in \Pi_s - \{\varepsilon_s\}$ they set

$$M_s(\alpha) := \bigoplus_{\gamma \in T_{s-1}(\alpha^-) - \bigcup_{(\alpha, \beta) \in E[\Pi_s]} T_{s-1}(\beta^-)} \phi(\gamma a) \otimes M_{s-1}(\gamma) \quad (35)$$

where $a = \alpha_{s:s}$ is the last letter of α and $T_{s-1}(\beta) = \{\gamma \mid \gamma = *\beta, \gamma \in \Pi_{s-1}\}$ for $\beta \in \Pi_{s-1}$ is the set of patterns in the branch of $G[\Pi_{s-1}]$ rooted at β . It can be shown that updates (34) and (35) are equivalent: they need exactly the same number of additions (and the same number of multiplications, if $\phi(\gamma a)$ in eq. (35) is replaced with $\phi(\alpha)$ and moved before the sum).

6.1. Proof of Theorem 9

To prove the correctness, we need to show that eq. (34) holds for each $\alpha \in \hat{\Pi}_s$.

Lemma 10. *For any $\alpha \in \hat{\Pi}_s$ there holds*

$$\phi(\alpha) \otimes M_{s-1}(\alpha^-) = \bigoplus_{x \in \mathcal{X}_s(\alpha; \hat{\Pi}_s)} f(x) \quad (36)$$

Proof. For $\alpha = \varepsilon_s$ the claim is trivial: we have $D^{s:s} \subseteq \hat{\Pi}_s$ (since $\varepsilon_{s-1} \in \Pi_{s-1}$), therefore $\mathcal{X}_s(\alpha; \hat{\Pi}_s) = \emptyset$ and the sum in (36) is \mathbb{O} . We thus assume that $\alpha \in \hat{\Pi}_s - \{\varepsilon_s\}$. Using definition (33), it can be checked that the mapping $x \mapsto x^-$ is a bijection $\mathcal{X}_s(\alpha; \hat{\Pi}_s) \rightarrow \mathcal{X}_{s-1}(\alpha^-; \Pi_{s-1})$. Consider $x \in \mathcal{X}_s(\alpha; \hat{\Pi}_s)$. We claim that if $x = *\gamma$ and $\gamma \in \Pi^\circ$ then $|\gamma| \leq |\alpha|$. Indeed, we have $\gamma \in \hat{\Pi}_s$ (since $\Pi_s^\circ \subseteq \hat{\Pi}_s$), and so if $|\gamma| > |\alpha|$ then $x \notin \mathcal{X}_s(\alpha; \hat{\Pi}_s)$ - a contradiction.

Using the claim, we conclude that $\phi(\alpha) \otimes f(x^-) = f(x)$. This implies the lemma. \square

The fact $\Pi_s \subseteq \hat{\Pi}_s$ implies the following characterization of $\mathcal{X}_s(\alpha; \Pi_s)$ for $\alpha \in \hat{\Pi}_s$:

$$\mathcal{X}_s(\alpha; \Pi_s) = \left\{ x \in \mathcal{X}_s(\alpha) \mid \begin{array}{l} x \neq *\beta \text{ for any } \beta \text{ in the subtree} \\ \text{of } \alpha \text{ in } G[\hat{\Pi}_s] \text{ with } \beta \in \Pi_s, \beta \neq \alpha \end{array} \right\}$$

where the subtree of α in $G[\hat{\Pi}_s]$ is defined as the set of descendants of α in $G[\hat{\Pi}_s]$ (including α).

Now it becomes clear that $\mathcal{X}_s(\alpha; \hat{\Pi}_s) \subseteq \mathcal{X}_s(\alpha; \Pi_s)$ and $\mathcal{X}_s(\alpha; \Pi_s) - \mathcal{X}_s(\alpha; \hat{\Pi}_s)$ equals the set of partial labelings $x \in \mathcal{X}_s(\alpha)$ such that

- x ends with $\beta \in \hat{\Pi}_s - \Pi_s$, $(\alpha, \beta) \in E[\hat{\Pi}_s]$, and
- x does not end with any pattern $\gamma \in \Pi_s$ from the subtree of β in $G[\hat{\Pi}_s]$.

It is easy to check that the last set of partial labelings equals $\mathcal{X}_s(\beta; \Pi_s)$, and such sets for different β 's are disjoint. We showed that $\mathcal{X}_s(\alpha; \Pi_s)$ is a disjoint union of sets $\mathcal{X}_s(\alpha; \hat{\Pi}_s)$ and $\mathcal{X}_s(\beta; \Pi_s)$ for $(\alpha, \beta) \in E[\hat{\Pi}_s]$, $\beta \notin \Pi_s$. This fact together with Lemma 10 implies eq. (34).

7. General case: $O(nP \log P)$ algorithm

In the previous section we presented an algorithm for a general commutative semiring with complexity $O(nP' |D|)$. In some applications the size of the input alphabet can be very large (e.g. hundreds or thousands), so the technique may be very costly. Below we present a more complicated version with complexity

$O(nP \log P)$. If $(R, \oplus, \otimes) = (\overline{\mathbb{R}}, \min, +)$ then this can be reduced to $O(nP \log(\ell_{\max} + 1))$ using the algorithm for *Range Minimum Queries* by (Berkman & Vishkin, 1993). We assume that $D \subseteq \Gamma$.

We will use the same definitions of sets Π_s and $\hat{\Pi}_s$ as in the previous section, and the same interpretation of messages $M_s(\alpha)$ given by eq. (31). We need to solve the following problem: given messages $M_{s-1}(\alpha)$ for $\alpha \in \Pi_{s-1}$, compute messages $M_s(\alpha)$ for $\alpha \in \Pi_s$.

Recall that in the previous section this was done by computing messages $M_s(\alpha)$ for patterns in the extended set $\hat{\Pi}_s$ of size $O(P' |D|)$. The idea of our modification is to compute these messages only for patterns in the set Σ_s where

$$\Sigma_s^\circ \subseteq \Sigma_s \subseteq \hat{\Pi}_s \quad \Sigma_s^\circ = \Pi_s \cup \Pi_s^\circ \quad |\Sigma_s| \leq 2|\Sigma_s^\circ|$$

Note that $|\Sigma_s^\circ| \leq P + 1$. Patterns in Σ_s will be called *special*. To define them, we will use the following notation for a node $\alpha \in \hat{\Pi}_s$:

- $\hat{\Phi}_s(\alpha)$ is the set of children of α in the tree $G[\hat{\Pi}_s]$.
- $\hat{T}_s(\alpha)$ is the set of nodes in the subtree of $G[\hat{\Pi}_s]$ rooted at α . We have $\alpha \in \hat{T}_s(\alpha) \subseteq \hat{\Pi}_s$.

We now define set Σ_s as follows: pattern $\alpha \in \hat{\Pi}_s$ is special if either (i) $\alpha \in \Sigma_s^\circ$, or (ii) α has at least two children $\beta_1, \beta_2 \in \hat{\Phi}_s(\alpha)$ such that subtree $\hat{T}_s(\beta_i)$ for $i \in \{1, 2\}$ contains a pattern from Σ_s° , i.e. $\hat{T}_s(\beta_i) \cap \Sigma_s^\circ \neq \emptyset$.

The set of remaining patterns $\hat{\Pi}_s - \Sigma_s$ will be split into two sets \mathcal{A}_s and \mathcal{B}_s as follows:

- \mathcal{A}_s is the set of patterns $\alpha \in \hat{\Pi}_s - \Sigma_s^\circ$ such that subtree $\hat{T}_s(\alpha)$ does not contain patterns from Σ_s° .
- \mathcal{B}_s is the set of patterns $\alpha \in \hat{\Pi}_s - \Sigma_s^\circ$ such that α has exactly one child β in $G[\hat{\Pi}_s]$ for which $\hat{T}_s(\beta) \cap \Sigma_s^\circ \neq \emptyset$.

Clearly, $\hat{\Pi}_s$ is a disjoint union of \mathcal{A}_s , \mathcal{B}_s and Σ_s .

Consider a node $\alpha \in \mathcal{B}_s$. From the definition, α has exactly one link to a child in $G[\hat{\Pi}_s]$ that belongs to $\Sigma_s \cup \Sigma_s^\circ$. If this child does not belong to Σ_s , then it belongs to \mathcal{B}_s and the same argument can be repeated for it. By following such links we eventually get to a node in Σ_s ; the first such node will be denoted as α_\downarrow .

We will need two more definitions. For an index t and patterns α, β ending at position t with $\beta = +\alpha$ we denote

$$W_t(\alpha) = \bigoplus_{x \in \mathcal{X}_t(\alpha)} f(x) \quad (37a)$$

$$V_t(\alpha, \beta) = \bigoplus_{x \in \mathcal{X}_t(\alpha) - \mathcal{X}_t(\beta)} f(x) \quad (37b)$$

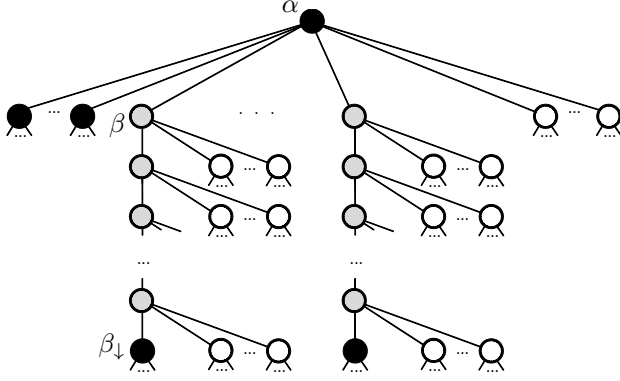


Figure 2. Structure of the subtree of $G[\widehat{\Pi}_s]$ rooted at a node $\alpha \in \Sigma_s$. White circles represent nodes in \mathcal{A}_s (so all their children are also white), gray circles - nodes in \mathcal{B}_s , and black circles - nodes in Σ_s . Note, if $\beta \in \mathcal{B}_s$ is a child of α then $(\alpha, \beta_{\downarrow}) \in E[\Sigma_s]$.

We can now formulate the structure of the algorithm.

Algorithm 5 Computing $Z = \bigoplus_{x \in D^{1:n}} f(x)$

- 1: initialize messages: set $M_0(\varepsilon) := \mathbb{O}$
- 2: for each $s = 1, \dots, n$ traverse nodes $\alpha \in \Sigma_s$ of tree $G[\Sigma_s]$ starting from the leaves and set

$$M_s(\alpha) := \phi(\alpha) \otimes [M_{s-1}(\alpha^-) \oplus A_s(\alpha) \oplus B_s(\alpha)] \oplus \bigoplus_{(\alpha, \beta) \in E[\Sigma_s], \beta \notin \Pi_s} M_s(\beta) \quad (38)$$

where

$$A_s(\alpha) = \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{A}_s} W_{s-1}(\beta^-) \quad (39a)$$

$$B_s(\alpha) = \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{B}_s} V_{s-1}(\beta^-, \beta_{\downarrow}^-) \quad (39b)$$

If $\alpha = \varepsilon_s$ then use $M_{s-1}(\alpha^-) := \mathbb{O}$

- 3: return $Z := \bigoplus_{\alpha \in \Pi_n} M_n(\alpha)$
-

To fully specify the algorithm, we still need to describe how we compute quantities $A_s(\alpha)$ and $B_s(\alpha)$ defined by eq. (39a) and (39b). This is addressed by the theorem below.

Theorem 11. (a) *Algorithm 5 is correct.*
 (b) *There holds $|\Sigma_s| \leq 2|\Sigma_s^{\circ}| - 1 \leq 2P + 1$.*
 (c) *Let h be the maximum depth of tree $G[\Pi_{s-1}]$. (Note, $h \leq \ell_{\max} + 1$.) With an $O(P' \log h)$ preprocessing, values $V_{s-1}(\alpha, \beta)$ for any $\alpha, \beta \in \Pi_{s-1}$ with $\beta = +\alpha$ can be computed in $O(\log h)$ time.*
 (d) *Values $A_s(\alpha)$ for all $\alpha \in \Sigma_s$ can be computed in $O(P \log P)$ time, or in $O(P)$ time when $(R, \oplus, \otimes) =$*

$(\mathbb{R}, \min, +)$.

Clearly, the theorem implies that the algorithm can be implemented in $O(nP \log P)$ time, or in $O(nP \log(\ell_{\max} + 1))$ time when $(R, \oplus, \otimes) = (\mathbb{R}, \min, +)$. To see this, observe that the sum in (39b) is effectively over a subset of children of α in the tree $G[\Sigma_s]$ (see Fig. 2), and this tree has size $O(P)$.

Before presenting the proof, let us make a few remarks. It can be easily checked that graph $G[\widehat{\Pi}_s]$ has the following structure: the root ε_s has $|D|$ children, and for each child $c \in D^{s:s} \subseteq \widehat{\Pi}_s$ the subtree of $G[\widehat{\Pi}_s]$ rooted at c is isomorphic to the tree $G[\Pi_{s-1}]$. The isomorphism $\widehat{T}_s(c) \rightarrow \Pi_{s-1}$ is given by the mapping $\alpha \mapsto \alpha^-$.

For nodes $\alpha, \beta \in \Pi_{s-1}$ with $\beta = +\alpha$ we denote $\mathcal{P}_{s-1}(\alpha, \beta)$ to be the unique path from α to β in $G[\Pi_{s-1}]$ (treated as a set of edges in $E[\Pi_{s-1}]$). Analogously, for nodes $\alpha, \beta \in \widehat{\Pi}_s$ with $\beta = +\alpha$ let $\widehat{\mathcal{P}}_s(\alpha, \beta)$ be the unique path from α to β in $G[\widehat{\Pi}_s]$. It follows from the previous paragraph that if $\alpha \neq \varepsilon_s$ then path $\widehat{\mathcal{P}}_s(\alpha, \beta)$ is isomorphic to the path $\mathcal{P}_{s-1}(\alpha^-, \beta^-)$.

7.1. Proof of Theorem 11(a)

The statement is equivalent to the correctness of (38). Let us first divide the sum over β in the last expression (38) into two parts: nodes β that belong to $\widehat{\Phi}(\alpha)$ and those that do not:

$$\bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap (\Sigma_s - \Pi_s)} M_s(\beta) \oplus \bigoplus_{(\alpha, \beta) \in E[\Sigma_s], \beta \notin \Pi_s \cup \Phi_s(\alpha)} M_s(\beta) \quad (40)$$

It is easy to see that the second sum can be written as $\sum_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{B}_s} \overline{M}_s(\beta_{\downarrow})$ (see Fig. 2), where

$$\overline{M}_s(\gamma) = \begin{cases} M_s(\gamma) & \text{if } \gamma \notin \Pi_s \\ \mathbb{O} & \text{otherwise} \end{cases} \quad (41)$$

Using the distributive law, we can rewrite (38) as

$$\begin{aligned} M_s(\alpha) &:= [\phi(\alpha) \otimes M_{s-1}(\alpha^-)] \\ &\oplus \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{A}_s} [\phi(\alpha) \otimes W_{s-1}(\beta^-)] \\ &\oplus \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{B}_s} [\phi(\alpha) \otimes V_{s-1}(\beta^-, \beta_{\downarrow}^-)] \oplus \overline{M}_s(\beta_{\downarrow}) \\ &\oplus \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap (\Sigma_s - \Pi_s)} M_s(\beta) \end{aligned} \quad (42)$$

We will use eq. (34) (for which the correctness is already proved) for the case when $\alpha \in \Sigma_s$. We can

rewrite it as follows (we use the fact that $\mathcal{A}_s \cap \Pi_s = \mathcal{B}_s \cap \Pi_s = \emptyset$):

$$M_s(\alpha) := [\phi(\alpha) \otimes M_{s-1}(\alpha^-)] \oplus \bigoplus_{\beta \in \widehat{\Pi}_s(\alpha) \cap \mathcal{A}_s} M_s(\beta) \\ \oplus \bigoplus_{\beta \in \widehat{\Pi}_s(\alpha) \cap \mathcal{B}_s} M_s(\beta) \oplus \bigoplus_{\beta \in \widehat{\Pi}_s(\alpha) \cap (\Sigma_s - \Pi_s)} M_s(\beta) \quad (43)$$

The first and the last terms of the sum in (43) equal to that of the sum in (42). The lemma below implies that the same holds for the second and third terms, thus proving the correctness of eq. (42).

Lemma 12. (a) For any $\alpha \in \mathcal{A}_s$ there holds

$$M_s(\alpha) = \phi(\alpha) \otimes W_{s-1}(\alpha^-) \quad (44)$$

(b) For any $\alpha \in \mathcal{B}_s$ there holds

$$M_s(\alpha) = [\phi(\alpha) \otimes V_{s-1}(\alpha^-, \alpha_\downarrow^-)] \oplus \overline{M}_s(\alpha_\downarrow) \quad (45)$$

Proof. Part (a) Suppose that $\alpha \in \mathcal{A}_s$. This means that there are no patterns $\beta \in \Pi_s^\circ$ of the form $\beta = +\alpha$ (recall that $\Pi_s^\circ \subseteq \Sigma_s^\circ$). This in turn implies that $M_s(\alpha) = W_s(\alpha)$. This also implies that for any $x \in \mathcal{X}_s(\alpha)$ there holds $f(x) = f(x^-)$, and consequently $W_s(\alpha) = \phi(\alpha) \otimes W_{s-1}(\alpha^-)$.

Part (b) Suppose that $\alpha \in \mathcal{B}_s$. The definition of \mathcal{B}_s implies that set $\widehat{T}_s(\alpha) - \widehat{T}_s(\alpha_\downarrow)$ does not contain nodes in Π_s or in Π_s° . Using this fact and the definition of sets $\mathcal{X}_s(\cdot)$, $\mathcal{X}_s(\cdot; \cdot)$ we get the following.

- (i) If $\alpha_\downarrow \in \Pi_s$ then $\mathcal{X}_s(\alpha; \Pi_s) = \mathcal{X}_s(\alpha) - \mathcal{X}_s(\alpha_\downarrow)$.
- (ii) If $\alpha_\downarrow \notin \Pi_s$ then $\mathcal{X}_s(\alpha; \Pi_s)$ is a disjoint union of $\mathcal{X}_s(\alpha) - \mathcal{X}_s(\alpha_\downarrow)$ and $\mathcal{X}_s(\alpha_\downarrow; \Pi_s)$.
- (iii) Partial labeling $x \in \mathcal{X}_s(\alpha) - \mathcal{X}_s(\alpha_\downarrow)$ cannot end with a pattern $\beta = +\alpha \in \Pi_s^\circ$. (If such β exists then from the fact above we get $\beta \in \widehat{T}_s(\alpha_\downarrow)$, i.e. $\beta = *\alpha_\downarrow$ and $x \in \mathcal{X}_s(\alpha_\downarrow)$ - a contradiction.) Therefore, for such x we have $f(x) = \phi(\alpha) \otimes f(x^-)$. This implies that $\sum_{x \in \mathcal{X}_s(\alpha) - \mathcal{X}_s(\alpha_\downarrow)} f(x) = \phi(\alpha) \otimes V_{s-1}(\alpha^-, \alpha_\downarrow^-)$.

Recall that $M_s(\alpha) = \sum_{x \in \mathcal{X}_s(\alpha; \Pi_s)} f(x)$. Using this fact and properties (i)-(iii), we conclude that (45) holds in each of the two cases ($\alpha_\downarrow \in \Pi_s$ and $\alpha_\downarrow \notin \Pi_s$). \square

7.2. Proof of Theorem 11(b)

For a node $\alpha \in \widehat{\Pi}_s$ we denote $T_s^\circ(\alpha) = \widehat{T}_s(\alpha) \cap \Sigma_s^\circ$.

Let us consider the process of a breadth-first search in the tree $G[\widehat{\Pi}_s]$ starting from the root. At each step we will keep a certain set of nodes of the tree (which we

call active nodes), and the transition to the next step is made by choosing one of the active nodes and replacing it with its children. The process stops when the set of active nodes becomes equal to the set of the leaves of the tree. To each step of the process we correspond a partition of the set Σ_s° . The partition is defined by the following rule: if $\alpha_1, \dots, \alpha_k$ are active nodes, then the partition is $\Sigma_s^\circ = \bigcup_{\ell=1}^k T_s^\circ(\alpha_\ell) \cup \bigcup_{\alpha \in \Sigma_s^\circ, \alpha \notin T_s^\circ(\alpha_\ell), \ell=1, \dots, k} \{\alpha\}$.

Let us denote the partition at step t as D_t .

Partitions of the set Π_s is a poset with respect to the natural order defined as follows: $\{S_i\}_{i \in I} \leq \{P_j\}_{j \in J}$ if for any $i \in I$ there is $j \in J$ such that $S_i \subseteq P_j$. It is easy to see that $D_0 \geq D_1 \geq D_2 \geq \dots$. Moreover, if a chosen active node α at step t is a special one and does not belong to Σ_s° then $D_t > D_{t+1}$. Indeed, there are at least two children of α , denoted as α_1 and α_2 , such that sets $T_s^\circ(\alpha_1)$ and $T_s^\circ(\alpha_2)$ are nonempty (by the definition of a special node). As step $t+1$ these sets are separate components of partition D_{t+1} , but at step t these sets still belong to the same component of D_t ; thus, $D_t \neq D_{t+1}$.

We know that the length of a chain $D_{t_1} > D_{t_2} > \dots$ cannot exceed $|\Sigma_s^\circ|$. We conclude that the number of special patterns that do not belong to Σ_s° is bounded by $|\Sigma_s^\circ| - 1$; this implies Theorem 11(b).

7.3. Proof of Theorem 11(c)

For brevity denote $t = s - 1$, and define a set of pairs

$$J = \{(\alpha, \beta) \mid \alpha, \beta \in \Pi_t, \beta \text{ is a strict descendant of } \alpha \text{ in } G[\Pi_t]\}$$

Note that $E[\Pi_t] \subseteq J$. In this section we describe a $O(P' \log h)$ preprocessing which will allow computing values $V_t(\alpha, \beta)$ for any $(\alpha, \beta) \in J$ in $O(\log h)$ time. The procedure will be based on the following observation; it follows trivially from the definition (37b).

Lemma 13. For any $(\alpha, \beta) \in J$ there holds

$$V_t(\alpha, \beta) = \bigoplus_{(\alpha', \beta') \in \mathcal{P}_t(\alpha, \beta)} V_t(\alpha', \beta') \quad (46)$$

During preprocessing we will compute values $V_t(\alpha, \beta)$ for pairs (α, β) in a certain set $\tilde{J} \subseteq J$ of size $O(P' \log h)$. In order to define \tilde{J} , we need some notation. For a pattern $\alpha \in \Pi_t$ let $h_\alpha = |\mathcal{P}_t(\varepsilon_t, \alpha)|$ be the height of α in $G[\Pi_t]$, and for an integer $d \in [0, h_\alpha]$ let $\alpha^{\uparrow d}$ be the node of tree $G[\Pi_t]$ obtained from α by taking d steps towards the root. We now define

$$\tilde{J} = \{(\alpha^{\uparrow d}, \alpha) \mid \alpha \in \Pi_t, d \in [0, h_\alpha], d = 2^r \text{ for } r \in \mathbb{Z}_{\geq 0}\}$$

The preprocessing will consist of 3 steps.

Step 1: compute values $W_t(\alpha)$ for all $\alpha \in \Pi_t$. We do it by traversing nodes $\alpha \in \Pi_t$ of tree $G[\Pi_t]$ and setting

$$W_t(\alpha) := M_t(\alpha) \oplus \bigoplus_{(\alpha, \beta) \in E[\Pi_t]} W_t(\beta) \quad (47)$$

This takes $O(P')$ time.

Step 2: go through $\alpha \in \Pi_t$ and compute values $V_t(\alpha, \beta)$ for all $\beta \in \Phi_t(\alpha)$ where $\Phi_t(\alpha)$ is the set of children of α in $G[\Pi_t]$.

A naive way is to use the formula

$$V_t(\alpha, \beta) = M_t(\alpha) \oplus \bigoplus_{\gamma \in \Phi_t(\alpha) - \{\beta\}} W_t(\gamma)$$

for all $\beta \in \Gamma_t(\alpha)$; however, this would take $O(k^2)$ time where $k = |\Phi_t(\alpha)|$. Instead, we do the following. Let us order patterns in $\Phi_t(\alpha)$ arbitrarily: $\Phi_t(\alpha) = \{\beta_1, \dots, \beta_k\}$. For $i \in [1, k]$ denote

$$\vec{S}_i = \bigoplus_{j=1}^{i-1} W_t(\beta_j) \quad \overleftarrow{S}_i = \bigoplus_{j=i+1}^k W_t(\beta_j)$$

We compute these values in $O(k)$ time by setting $\vec{S}_1 := \emptyset$, $\overleftarrow{S}_k := \emptyset$ and then using recursions

$$\vec{S}_{i+1} := \vec{S}_i \oplus W_t(\beta_i) \quad \overleftarrow{S}_{i-1} := \overleftarrow{S}_i \oplus W_t(\beta_i)$$

After that we set

$$V_t(\alpha, \beta_i) := M_t(\alpha) \oplus \vec{S}_i \oplus \overleftarrow{S}_i$$

For a given $\alpha \in \Pi_t$ the procedure takes $O(|\Phi_t(\alpha)|)$ time, and thus for all $\alpha \in \Pi_t$ it takes $O(P')$ time.

We now have values $V_t(\alpha, \beta)$ for all $(\alpha, \beta) \in E[\Pi_t]$.

Step 3: compute values $V_t(\alpha, \beta)$ for all $(\alpha, \beta) \in \tilde{J}$ using the recursion

$$V_t(\alpha^{\uparrow 2d}, \alpha) := V_t(\alpha^{\uparrow 2d}, \alpha^{\uparrow d}) + V_t(\alpha^{\uparrow d}, \alpha)$$

for $d = 2^0, 2^1, \dots, 2^r, \dots$ and $(\alpha^{\uparrow 2d}, \alpha) \in \tilde{J}$.

Evaluating queries for $(\alpha, \beta) \in J$ We showed how to compute values $V_t(\alpha, \beta)$ for $(\alpha, \beta) \in \tilde{J}$ in time $O(P' \log h)$; let us now describe how to compute value $V_t(\alpha, \beta)$ for a given $(\alpha, \beta) \in J$ in time $O(\log h)$. Let us construct a sequence β_0, β_1, \dots , as follows: $\beta_0 = \beta$, and for $i \geq 0$ let $\beta_{i+1} = \beta_i^{\uparrow d}$ where d is the maximum value such that $d = 2^r$ for $r \in \mathbb{Z}_{\geq 0}$ and $\beta_i^{\uparrow d}$ is still a descendant of α . We stop when we get $\beta_k = \alpha$; clearly, this happens after $k = O(\log h)$ steps. We now set

$$V_t(\alpha, \beta) := \bigoplus_{i=0}^{k-1} V_t(\beta_{i+1}, \beta_i)$$

7.4. Proof of Theorem 11(d)

We will consider the general case of a commutative semiring and the case when $(R, \oplus, \otimes) = (\mathbb{R}, \min, +)$; the latter will be called the *MAP case*.

Let d_α for $\alpha \in \Pi_{s-1}$ be the number of children of α in $G[\Pi_{s-1}]$, $d_{\max} = \max_{\alpha \in \Pi_{s-1}} d_\alpha$, and \tilde{d}_α for $\alpha \in \Sigma_s$ be the number of children of α in $G[\Sigma_s]$. We will present a $O(\sum_{\alpha \in \Pi_{s-1}} d_\alpha \log d_\alpha)$ preprocessing technique that will allow computing value $A_s(\alpha)$ for $\alpha \in \Sigma_s - \{\varepsilon_s\}$ in time $O((\tilde{d}_\alpha + 1) \log d_{\alpha-})$. The resulting complexity will be

$$O(\sum_{\alpha \in \Pi_{s-1}} d_\alpha \log d_{\max}) + \sum_{\alpha \in \Sigma_s} O((\tilde{d}_\alpha + 1) \log d_{\max}) = O(P \log d_{\max})$$

In the *MAP case* (i.e. when $(R, \oplus, \otimes) = (\mathbb{R}, \min, +)$) we will present a faster solution. Namely, the preprocessing will take $O(\sum_{\alpha \in \Pi_{s-1}} d_\alpha) = O(P')$ time, and computing value $A_s(\alpha)$ for $\alpha \in \Sigma_s - \{\varepsilon_s\}$ will take $O(\tilde{d}_\alpha + 1)$ time, leading to the overall complexity $O(P)$. For that we will use the *Range Minimum Query (RMQ)* problem which is defined as follows: given N numbers z_1, \dots, z_N , compute $\min_{k \in I} z_k$ for a given interval $I = [i, j] \subseteq [1, N]$. It is known (Berkman & Vishkin, 1993) that with an $O(N)$ preprocessing each query for can be answered in $O(1)$ time per interval.

As in the previous section, we denote $t = s - 1$. We assume that for each $\alpha \in \Pi_t$ we already have values $W_t(\alpha)$ for all $\alpha \in \Pi_t$ (they were computed in the previous section).

Preprocessing Consider $\alpha \in \Pi_t$, and let us fix an ordering of children of α : $\Phi_t(\alpha) = \{\beta_1, \dots, \beta_d\}$ where $d = d_\alpha$. For an interval $I \subseteq [1, d]$ we denote

$$S_I(\alpha) = \bigoplus_{i \in I} W_t(\beta_i) \quad (48)$$

The goal of preprocessing is to build a data structure that we will allow an efficient computation of $S_I(\alpha)$ for any given interval I .

In the *MAP case* we simply run the preprocessing of (Berkman & Vishkin, 1993) for the sequence $W_t(\beta_1), \dots, W_t(\beta_d)$; this takes $O(d)$ time. Value $S_I(\alpha)$ for an interval I can then be computed in $O(1)$ time.

In the general case we do the following. Define a set of intervals

$$J_d = \{[i, j] \subseteq [1, d] \mid j - i = 2^r - 1, r \in \mathbb{Z}_{\geq 0}\}$$

Note that $|J_d| = O(d \log d)$. We compute quantities $S_I(\alpha)$ for all $I \in J_d$. This can be done in $O(d \log d)$ time by setting $S_{[i, i]}(\alpha) := W_t(\beta_i)$ for $i \in [1, d]$ and

then using recursions

$$S_{[i,i+2\delta-1]}(\alpha) = S_{[i,i+\delta-1]}(\alpha) \oplus S_{[i+\delta,i+2\delta-1]}(\alpha)$$

for $\delta = 2^0, 2^1, \dots$

Value $S_I(\alpha)$ for an interval I can now be computed in $O(\log d)$ time. Indeed, we can represent I as a disjoint union of $m = O(\log d)$ intervals from J_d : $I = \bigcup_{i=1}^m I_i$ with $I_i \in J_d$. We can then use the formula

$$S_I(\alpha) = \bigoplus_{i=1}^m S_{I_i}(\alpha) \quad (49)$$

Computing $A_s(\alpha)$ for $\alpha \in \Sigma_s - \{\varepsilon_s\}$ Denote $d = d_{\alpha^-}$. As discussed earlier, $\widehat{\Phi}_s(\alpha)$ (the set of children of α in $G[\widehat{\Pi}_s]$) is isomorphic to $\Phi_{s-1}(\alpha^-)$. Let $\widehat{\Gamma}_s(\alpha) = \{\beta_1, \dots, \beta_d\}$ be the ordering of patterns in $\widehat{\Phi}_s(\alpha)$ such that $\beta_1^-, \dots, \beta_d^-$ is the ordering of patterns in $\Phi_{s-1}(\alpha^-)$ chosen in the preprocessing step. We need to compute

$$A_s(\alpha) = \bigoplus_{i \in J} W_{s-1}(\beta_i^-), \quad J \triangleq \{i \in [1, d] \mid \beta_i \in \mathcal{A}_s\}$$

Denote $\bar{J} = [1, d] - J$. We can represent J as a disjoint union of at most $|\bar{J}| + 1$ intervals I_1, \dots, I_m where $I_i \subseteq [1, p]$. Clearly, $|\bar{J}| = \tilde{d}_\alpha$ (see Fig. 2), and so $m \leq \tilde{d}_\alpha + 1$. We can write

$$A_s(\alpha) = \bigoplus_{i=1}^m S_{I_i}(\alpha^-) \quad (50)$$

As discussed above, each value $S_{I_i}(\alpha^-)$ can be computed in $O(1)$ time in the MAP case and in $O(\log d)$ in the general case. Thus, computing $A_s(\alpha)$ takes respectively $O(\tilde{d}_\alpha + 1)$ and $O((\tilde{d}_\alpha + 1) \log d_{\alpha^-})$ time, as desired.

8. MAP for non-positive costs

In this section we assume that $(R, \oplus, \otimes) = (\mathbb{R}, \min, +)$ and $c_\alpha \leq 0$ for all $\alpha \in \Pi^\circ$. (Komodakis & Paragios, 2009) gave an algorithm that makes $\Theta(nP)$ comparisons and $\Theta(nP)$ additions. We will present a modification that makes only $O(n|I(\Gamma)|)$ comparisons. The number of additions in general will still be $O(nP)$, but we will show that in certain scenarios it can be reduced using a Fast Fourier Transform (FFT).

We will assume that Γ contains at least one word α with $|\alpha| = 1$ (it can always be added if needed).

As usual, we first select a set of patterns Π with $\Pi_0 = \{\varepsilon_0\}$; this step will be described later. For a pattern $\alpha \in \Pi$ let α^\leftarrow be the longest proper prefix of α that

is in Π ($\alpha = \alpha^\leftarrow +$, $\alpha^\leftarrow \in \Pi$). If Π does not contains proper prefixes of α then α^\leftarrow is undefined.

We can now present the algorithm.

Algorithm 6 Computing $Z = \min_{x \in D^{1:n}} f(x)$ (if $c_\alpha \leq 0$)

- 1: initialize messages: set $M_0(\varepsilon_0) := 0$
- 2: for each $s = 1, \dots, n$ traverse nodes $\alpha \in \Pi_s$ of forest $G[\Pi_s]$ starting from the leaves and set

$$M_s(\alpha) := \min\{M_p(\alpha^\leftarrow) + \psi(\alpha), \min_{(\alpha, \beta) \in E[\Pi_s]} M_s(\beta)\} \quad (51)$$

where p is the end position of α^\leftarrow ($\alpha^\leftarrow = ([\cdot, p], \cdot)$) and $\psi(\alpha) = f(\alpha) - f(\alpha^\leftarrow)$. If α^\leftarrow is undefined then ignore the first expression in (51).

- 3: return $Z := \min_{\alpha \in \Pi_n} M_n(\alpha)$
-

Selecting Π It remains to specify how to choose set Π . For patterns α, β, γ we define

$$\langle \alpha | \beta | \gamma \rangle = \{u = +\beta + \mid \alpha\beta\gamma = *u*\} \quad (52)$$

Theorem 14. Suppose that $\Pi_0 = \{\varepsilon_0\}$ and set Π contains set

$$\tilde{\Pi} = \left\{ \beta \mid \begin{array}{l} \exists \text{ labeling } x\alpha\beta\gamma y \in D^{1:n} \text{ s.t.} \\ (a) \alpha\beta, \beta\gamma \in \Pi^\circ; (b) \langle x\alpha | \beta | \gamma y \rangle \cap \Pi^\circ = \emptyset \end{array} \right\} \quad (53)$$

Then Alg. 6 returns the correct value of $Z = \min_{x \in D^{1:n}} f(x)$.

A proof of this theorem is given in Sec. 8.2.

A simple valid option is to set $\Pi = \{\alpha \mid \exists \alpha*, *\alpha \in \Pi^\circ\}$. Computing set $\tilde{\Pi}$ is slightly more complicated, but can still be done in polynomial time for a given Π (we omit this procedure). In order to analyze set $\tilde{\Pi}$, let us define

$$I_\delta = \left\{ \beta \mid \begin{array}{l} \exists \text{ word } x\alpha\beta\gamma y \text{ with } |x| = |y| = \delta \\ \text{s.t. (a) } \alpha\beta, \beta\gamma \in \Gamma; (b) \langle x\alpha | \beta | \gamma y \rangle \cap \Gamma = \emptyset \end{array} \right\}$$

where set $\langle \cdot | \cdot | \cdot \rangle$ for words is defined similarly to (52):

$$\langle \alpha | \beta | \gamma \rangle = \{\hat{\alpha}\beta\hat{\gamma} \mid \alpha = *\hat{\alpha}, \gamma = \hat{\gamma}* \text{ and } \hat{\alpha}, \hat{\gamma} \neq \varepsilon\} \quad (54)$$

As δ increases, set I_δ monotonically shrinks, and stops changing after $\delta \geq \ell_{\max} = \max_{\alpha \in \Gamma} |\alpha|$. We denote this limit set as I_∞ , so that $I_\infty \subseteq I_0 \subseteq I(\Gamma) \cup \{\varepsilon\}$. It can be seen that $\tilde{\Pi}_s = \{([\cdot, s], \alpha) \mid \alpha \in I_\infty\}$ for all $s \in [\ell_{\max}, n - \ell_{\max} + 1]$.

Complexity Assume that we use $\Pi = \tilde{\Pi}$. The algorithm performs two types of operations: comparisons (to compute minima) and arithmetic operations (to compute the first expression in (51)). The number of comparisons does not exceed the total number of edges

in graphs $G[\Pi_s] = (\Pi_s, E[\Pi_s])$ for $s \in [1, n]$, which is smaller than the number of nodes (since graphs are forests). Thus, comparisons take $O(n|I_\infty|)$ time.

The time for arithmetic operations depends on how we compute quantities $f(\alpha)$. One possible approach is to use Lemma 1 for computing $f(\alpha)$ for all $\alpha \in \hat{\Pi}$ where $\hat{\Pi}$ is the set of prefixes of patterns in Π° (note that $\Pi \subseteq \hat{\Pi}$). We have $|\hat{\Pi}_s| \leq P + 1$, and therefore the resulting overall complexity is $O(nP)$. Next, we describe an alternative approach based on a Fast Fourier Transform.

8.1. Computing $f(\alpha)$ using FFT

For a word α and index $s \geq |\alpha|$ let α_s be the pattern $([s - |\alpha| + 1, s], \alpha)$. It is easy to see that

$$f(\alpha_s) = \sum_{\beta \in \Gamma} f_s(\alpha|\beta) \quad \text{where} \quad f_s(\alpha|\beta) = \sum_{t: \alpha_s = * \beta_t *} c_{\beta_t}$$

Lemma 15. *For fixed words α, β quantities $f_s(\alpha|\beta)$ for $s \in [1, n]$ can be computed in $O(n \log n)$ time.*

Proof. We assume that $|\alpha| \geq |\beta|$, otherwise the claim is trivial. Let $p = |\alpha| - |\beta| + 1$, and define sequences $a \in \mathbb{R}^{n-|\alpha|+1}$, $b \in \mathbb{R}^{n-|\beta|+1}$, $\lambda \in \{0, 1\}^p$ via

$$\begin{aligned} a_i &= f_{i+|\alpha|}(\alpha|\beta) & \forall i \in [1, n-|\alpha|] \\ b_j &= c_{([j, j+|\beta|-1], \beta)} & \forall j \in [1, n-|\beta|+1] \\ \lambda_k &= [\alpha_{k:k+|\beta|-1} = \beta] & \forall k \in [1, p] \end{aligned}$$

where $[\cdot]$ is the Iverson bracket. It can be checked that

$$a_i = \sum_{k=1}^p b_{i+k} \lambda_k \quad \forall i \in [1, n-|\alpha|]$$

Thus, a is the convolution of b and the reverse of λ . The convolution of such sequences can be computed in $O(n \log n)$ using a Fast Fourier Transform. \square

In practice this method can be useful for computing $f_s(\alpha|\beta)$ when $|\alpha| \gg |\beta|$. A natural way to do this is first choose a subset $S \subseteq \Gamma$ of words that are very often included as subwords in words from I_∞ (usually, this means $S = \{\beta \mid \beta \in \Gamma, |\beta| \leq \delta\}$ where δ is some threshold constant). Then computing $f_s(\alpha|\beta)$ for all $\alpha \in I_\infty$ and $\beta \in S$ will take $O(|I_\infty| \cdot |S| n \log n)$ time. For $\beta \in \Gamma - S$ quantities $f_s(\alpha|\beta)$ can be computed directly.

An example of using such an approach is the following theorem; it is proved by taking $\delta = 1$ and $S = D$.

Theorem 16. *Suppose $\Gamma = D \cup A$ where A consists of words of a fixed length ℓ . Then Algorithm 6 can be implemented in $O(|I_\infty| \cdot |D| n \log n)$ time.*

8.2. Proof of Theorem 14 (correctness)

First, let us prove that for all patterns $\alpha = ([\cdot, s], \cdot) \in \Pi$ there holds

$$M_s(\alpha) \geq \min_{x=* \alpha \in D^{1:s}} f(x) \quad (55)$$

We use induction on the order used in the algorithm. The base case $\alpha = \varepsilon_0$ is ensured by the initialization step. Consider pattern $\alpha \in \Pi - \{\varepsilon_0\}$. Suppose that α^\leftarrow is defined; let p be the end position of α^\leftarrow . Consider partial labeling $y = * \alpha^\leftarrow \in D^{1:p}$, and let x be its unique extension by $s - p$ letters ($x = y +$) such that $x = * \alpha$. Due to non-positivity of costs c_β we have

$$f(x) = f(y) + \psi(\alpha) + \sum_{\substack{\beta = ([i, j], \cdot) \in \Pi^\circ \\ x = * \beta * \\ i \leq s - |\alpha|, j > p}} c_\beta \leq f(y) + \psi(\alpha)$$

Applying the induction hypothesis and the inequality above yields the desired claim:

$$\begin{aligned} M_s(\alpha) &= \min\{M_p(\alpha^\leftarrow) + \psi(\alpha), \min_{(\alpha, \beta) \in E[\Pi_s]} M_s(\beta)\} \\ &\geq \min\{\min_{y=* \alpha^\leftarrow} f(y) + \psi(\alpha), \min_{(\alpha, \beta) \in E[\Pi_s]} \min_{x=* \beta} f(x)\} \\ &\geq \min\{\min_{x=* \alpha} f(x), \min_{x=* \alpha} f(x)\} = \min_{x=* \alpha} f(x) \end{aligned}$$

where in the equations above x always denotes a partial labeling in $D^{1:s}$ and y denotes a partial labeling in $D^{1:p}$. If α^\leftarrow is undefined then we can write similar inequalities but omitting expressions containing α^\leftarrow .

By applying the claim to patterns $\alpha = ([\cdot, n], \cdot) \in \Pi$ we conclude that $Z \geq \min_{x \in D^{1:n}} f(x)$. The remainder of this section is devoted to the proof of the reverse inequality: $Z \leq \min_{x \in D^{1:n}} f(x)$.

Let us fix $x^* \in \arg \min_{x \in D^{1:n}} f(x)$. Let

$$\Lambda = \{\alpha \in \Pi^\circ \mid x^* = * \alpha * \}$$

be the set of patterns present in x^* , and

$$\hat{\Lambda} = \{\alpha \in \Lambda \mid \text{there is no } \hat{\alpha} \in \Lambda - \{\alpha\} \text{ with } \hat{\alpha} = * \alpha * \}$$

be the set of *maximal* patterns in Λ . We can assume w.l.o.g. that for each $k \in [1, n]$ there exists $\alpha = ([i, j], \cdot) \in \hat{\Lambda}$ with $k \in [i, j]$. Indeed, if it is not the case for some k then we can modify x^* by replacing the k -th letter of x^* with some letter $c \in \Gamma \cap D$; this operation does not increase $f(x^*)$.

We define a total order \preceq on patterns $\alpha = ([i, j], \cdot) \in \hat{\Lambda}$ as the lexicographical order with components (i, j) (the first component is more significant).

Lemma 17. (a) For each pattern $\beta \in \hat{\Lambda}$ there holds $\beta \in \Pi$.

(b) Consider two consecutive patterns $\alpha_1 \prec \alpha_2$ in $\hat{\Lambda}$ with $\alpha_1 = ([i_1, j_1], \cdot)$, $\alpha_2 = ([i_2, j_2], \cdot)$, and let $\beta = x_{i_2:j_1}^*$ be the pattern at which they intersect. (By the assumption above, $i_2 \leq j_1 + 1$). There holds $\beta \in \Pi$.
 (c) For the patterns in (b), condition $M_{j_1}(\alpha_1) \leq f(x_{1:j_1}^*)$ implies $M_{j_2}(\alpha_2) \leq f(x_{1:j_2}^*)$.

Proof. Part (a) We can write labeling x^* as $x^* = x\alpha\beta\gamma y$ where patterns α, β are empty. Let us show that this choice satisfies conditions in (53). Condition (a) holds since $\alpha\beta = \beta\gamma = \beta \in \Pi^\circ$. Suppose that (b) does not hold, then there exists pattern $u = x_{k:\ell}^* \in \Pi^\circ$ with $u = +\beta+$. We have $u \in \Lambda$ and thus $\beta \notin \hat{\Lambda}$ - a contradiction. Therefore, $\beta \in \Pi$.

Part (b) The definitions of \preceq and $\hat{\Lambda}$ imply that $i_1 < i_2$. (If $i_1 = i_2$ then we must have $j_1 < j_2$, but then $\alpha_1 \notin \hat{\Lambda}$ - a contradiction.) There also holds $j_1 < j_2$ (otherwise we would have $\alpha_2 \notin \hat{\Lambda}$ - a contradiction). This means that we can write labeling x^* as $x^* = x\alpha\beta\gamma y$ where $\alpha\beta = \alpha_1$, $\beta\gamma = \alpha_2$ and the pattern intervals are as follows:

$$\begin{array}{cccccc} x & \alpha & \beta & \gamma & y \\ [1, i_1 - 1] & [i_1, i_2 - 1] & [i_2, j_1] & [j_1 + 1, j_2] & [j_2 + 1, n] \end{array}$$

Let us show that this choice satisfies conditions in (53). Condition (a) holds since $\alpha\beta = \alpha_1 \in \Pi^\circ$ and $\beta\gamma = \alpha_2 \in \Pi^\circ$. Suppose that (b) does not hold, then there exists pattern $u = x_{k:\ell}^* \in \Pi^\circ$ where $k < i_2$ and $\ell > j_1$. This means that $u \in \Lambda$.

From the definition of $\hat{\Lambda}$, there exists pattern $\hat{u} = x_{\hat{k}:\hat{\ell}}^* \in \hat{\Lambda}$ with $[k, \ell] \subseteq [\hat{k}, \hat{\ell}]$. To summarize, we have $\hat{k} \leq k < i_2$ and $j_1 < \ell \leq \hat{\ell}$.

If $\hat{k} \leq i_1$ then $[i_1, j_1] \subset [\hat{k}, \hat{\ell}]$ and thus $\alpha_1 \notin \hat{\Lambda}$ - a contradiction. Thus, there must hold $\hat{k} > i_1$. Similarly, we prove that $\hat{\ell} < j_2$. This implies that $\alpha_1 \prec \hat{u} \prec \alpha_2$, and therefore patterns α_1 and α_2 are not consecutive in $\hat{\Lambda}$ - a contradiction.

Part (c) β is a proper prefix of α that belongs to Π . Therefore, α^\leftarrow is defined. We can define a sequence of patterns $\beta_0 = \beta, \beta_1, \dots, \beta_m = \alpha_2$ with $\beta_k = ([i_2, s_k], \cdot)$, $s_0 = j_1 < s_1 < \dots < s_m = j_2$ respectively such that $\beta_k \in \Pi$ and $\beta_{k-1} = \beta_k^\leftarrow$ for $k \in [1, m]$. Let us prove by induction on k that $M_{s_k}(\beta_k) \leq f(x_{1:s_k}^*)$ for $k \in [0, m]$.

Let us first check the base of the induction. Since $\beta, \alpha_1 \in \Pi_{j_1}$ and $\alpha_1 = *\beta$, by the definition of graph $G[\Pi_{j_1}]$ there is a (unique) path $\gamma_0, \gamma_1, \dots, \gamma_r$ from $\gamma_0 = \beta$ to $\gamma_r = \alpha_1$ with $(\gamma_l, \gamma_{l+1}) \in E[\Pi_{j_1}]$. By eq. (51), $M_{j_1}(\gamma_l) \leq M_{j_1}(\gamma_{l+1})$. Therefore, $M_{j_1}(\beta) \leq$

$M_{j_1}(\alpha_1) \leq f(x_{1:j_1}^*)$ where the last inequality holds by the assumption of part (c). This establishes the base case.

Now suppose that the claim holds for $k - 1 \in [0, m - 1]$; let us prove it for k . Denote $p = s_{k-1}$ and $s = s_k$. Note, p is the index in step 2 of the algorithm during the processing of (s, β_k) . From eq. (51) and the induction hypothesis we get

$$M_s(\beta_k) \leq M_p(\beta_{k-1}) + \psi(\beta_k) \leq F_p(x_{1:p}^*) + \psi(\beta_k)$$

We will prove next that $f(x_{1:p}^*) + \psi(\beta_k) = f(x_{1:s}^*)$, thus completing the induction step. It suffices to show that there is no pattern $\gamma = x_{i:j}^*$ with $i < i_2$ and $p < j \leq s$.

Suppose on the contrary that such pattern exists. By the definition of $\hat{\Lambda}$ there exists pattern $\hat{\gamma} = ([\hat{i}, \hat{j}], \cdot) \in \hat{\Lambda}$ with $[i, j] \subseteq [\hat{i}, \hat{j}]$. We have $\hat{i} \leq i < i_2$ and $\hat{j} \geq j > p \geq s_0 = j_1$. These facts imply that $\alpha_1 \prec \hat{\gamma} \prec \alpha_2$, contradicting the assumption that α_1 and α_2 are consecutive patterns in $\hat{\Lambda}$. \square

Lemma 17 implies the main claim.

Corollary 18. For each $\alpha = ([i, j], \cdot) \in \hat{\Lambda}$ there holds $M_j(\alpha) \leq f(x_{1:j}^*)$, and therefore $Z \leq f(x^*)$.

Proof. We use induction on the total order \preceq . The lowest pattern $\alpha \in \hat{\Lambda}$ starts at position 1; for such pattern the claim follows by inspecting Algorithm 6. The induction step follows from Lemma 17(c). \square

Acknowledgements

We thank Herbert Edelsbrunner for helpful discussions.

References

- Berkman, Omer and Vishkin, Uzi. Recursive star-tree parallel data structure. *SIAM Journal on Computing*, 22(2):221–242, 1993.
- Komodakis, N. and Paragios, N. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.
- Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.
- Nguyen, Viet Cuong, Ye, Nan, Lee, Wee Sun, and Chieu, Hai Leong. Semi-Markov conditional random field with high-order features. In *ICML 2011 Structured Sparsity: Learning and Inference Workshop*, 2011.

- Qian, Xian, Jiang, Xiaoqian, Zhang, Qi, Huang, Xuan-jing, and Wu, Lide. Sparse higher order conditional random fields for improved sequence labeling. In *ICML*, 2009.
- Rother, C., Kohli, P., Feng, W., and Jia, J. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, 2009.
- Sarawagi, S. and Cohen, W. Semi-Markov conditional random fields for information extraction. In *NIPS*, 2004.
- Vose, M. D. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.
- Ye, Nan, Lee, Wee Sun, Chieu, Hai Leong, and Wu, Dan. Conditional random fields with high-order features for sequence labeling. In *NIPS*, 2009.